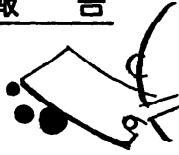


報告



パネル討論会

ソフトウェアの使いやすさ

昭和57年前期第24回全国大会報告

パネリスト

伊藤 正雄¹⁾、片岡 雅憲²⁾、木村 泉³⁾
司会 福田 善一⁴⁾

本日のパネル討論は、コンピュータの大量普及時代を迎えて、最も関心がよせられているテーマの一つ、「ソフトウェアの使いやすさ」についてである。パネリストとしては、利用者、生産者、教育・研究者のそれぞれの立場の方々をお願いした。

このテーマは、総論としては明確であるが、各論になるとなかなかむづかしいといえる。その理由は、利用者層とソフトウェアの種類が多岐にわたっており、使いやすさの測定尺度について一致した見解がなく、かつ定量的測定方法がないことによるといえよう。そこで、各パネリストのご意見をここにまとめておくべきかもしれないが、それは各パネリストの報告におまかせして、ここではこのテーマを討論するに当たって考えておきたいこと3点について述べさせて頂く。

1. なぜ使いやすさに関心がよせられるか

(1) システムの利用者と対象領域の広がり

今日システムの開発はコンピュータ部門が行い、利用はライン全員が対象となっている。つまり開発者と利用者の分離が進み、この事が急速にコンピュータ利用人口を増加させた。さらに TSS の普及とパーソナルコンピュータの登場が新しい形で利用者増加に拍車をかけている。利用者は単に増加しただけでなく多様化している。つまりあらゆる分野と階層の人がいて、その年齢、利用経験、使用頻度がさまざまである。

システムの対象領域もネットワークの拡大とパーソナルコンピュータやオフィスコンピュータの普及によって空間的広がりや質的深まりを示している。

(2) ソフトウェアの巨大化、複雑化

オペレーティングシステムを例にとると、その大い

さは数キロステップであったものが、今日では数メガステップになっている。このようなシステムの巨大化複雑化によりシステムのブラックボックス化が進んでいる。利用者はひとたびエラーが出たり例外処理が要求されると、どうしてよいかわからなくなるという不安感をつのらせており、こういう面からの使いやすさが要求されている。

(3) 誤操作による影響の深刻さ

使いにくさによる操作ミスがシステム障害を引き起こし、その場合の影響の大きさは益々深刻になっている。さらに故意による犯罪の要素も含めてシステムの安全性からソフトウェアの使いやすさが認識されている。

(4) ハードウェア技術の進歩と価格低下

かつて日本語は使いたくてもハードウェアが高価で使えなかった。今日 LSI やレーザ技術の進歩によりハードウェアは安くなり小型化して、ソフトウェアの充実と相俟って容易に使えるようになった。情報システムが日本語で扱えることは、使いやすさを大きく進展させることになる。このようにハードウェア技術の進歩が使いやすさを適正な価格で入手可能にしている。

(5) 価値感の変化

ソフトウェアの開発が単純で無味乾燥な作業では、もはや若人にも受け入れられなくなってきた。利用面でも CAD などのように創造活動への適用が重視されてきてソフトウェアに新しい要素が出てきている。さらに技術の進歩は民族的、文化的要素も使いやすさに反映し得る余地を与えつつある。

このような背景から、今日「ソフトウェアの使いやすさ」への関心が高まってきたといえよう。

2. 使いやすいとはどんなことか

† 日時 昭和57年3月22日, 15:00~17:00

場所 東京電機大学

1) 新日本製鉄(株), 2) 日立製作所, 3) 東京工業大学,

4) 富士通(株)

ソフトウェアの使いやすさを論じるために、その評価項目と定量的評価尺度が必要である。ここではエンドユーザ製品を対象として、トータルな使いやすさのうち製品共通の要素だけについて評価項目をあげる。

(1) 親しみやすさ (friendliness)

第一は利用者の助けになる機能で、構文の柔軟性、使い慣れた表現方法、メニューによるガイダンスの提供や HELP 機能である。第二は安全性で、操作履歴の保持と再現、誤りの早期検出と修正に役立つメッセージ出力など。第三は人の思考にそった手順で作業ができるなどの人間工学の要素。第四は個人への適応性で、利用者のレベルに応じたメニュー、コマンド、メッセージの用意、コマンド名の変更ができるなどの拡張性である。第五は親しみやすいインタフェースで、メニューによる項目選択ができるなど選択の容易性、同一コマンドなら応答時間が予測可能であるなど。

(2) わかりやすさ (understandability)

第一は習得の容易性で、とっつきやすい、習得時間が短い、忘れにくい、思い出す時間が短いなど。第二は理解しやすさで、明確で理解しやすいマニュアル、わかりやすいメッセージ、わかりやすい出力データなどである。

(3) 効率のよさ (efficiency)

第一は高い生産性で、仕事をする準備や実行時間が短く、大きな作業領域を必要とすると特殊な端末がいるなどの条件がつかない、またコンピュータ経費が少ないこと。第二は効率的なインタフェースで、ユーザの注意を促すのに視覚、聴覚、触覚の利用、コマンド入力の労力を少なくするとか、他の作業へ簡単にいききできること。第三は低い使用エラー率で、利用者の単純な操作ミスや論理ミスが起りにくい考慮、システム自身にバグが少ないことである。

(4) 寛容性 (ruggedness)

第一はエラーについての考慮で、ファイルの消去は利用者に確認とかログオフする前に作業ファイルを保存したかの確認、利用者に無関係のエラーメッセージは見せないこと。第二はシステムの保全で、プログラムやファイルやデータの保護、ファイルや作業の自動リカバリ機能の提供。第三は丈夫さで、エラーが起きた時でも処理が続けられること、万一システムダウンやアベンドしても止める前に次の行動がわかるメッセージの出力などである。

これらの評価項目へのウェイトづけは、評価する人の立場、コンピュータ経験、使用頻度、対象業務など

によって異なるので評価尺度は一樣ではない。

3. 対象範囲はどこまでとするか

ソフトウェアの使いやすさを検討する時、利用者、アプリケーション、技術の進歩、価値感の変化などのどこまでを対象とするかを設定する必要がある。

(1) 利用者

だれにとって使いやすいかという時の主体の対象範囲についてである。一番多いのが一般企業などいわゆるコンピュータユーザである。このユーザの中には、EDP 部門の人とエンドユーザがあり、エンドユーザには端末などを直接利用する人と、その仕事の責任者であるミドルマネジメントとトップマネジメントがある。つぎにコンピュータメカやソフトウェアハウスの人々、さらに教育者・研究者、行政の立場の人などがあげられる。

(2) アプリケーション

第一に一般企業のシステムである。この中にも定型処理中心の基幹システムと非定型処理のものがある。前者にはオーダエントリとか経理業務などと設計支援システム CAD などがあり、後者には意思決定支援システム MDS などがある。さらにパーソナルコンピュータを個人や個別部門に使うセクションシステムがある。第二に社会システムと呼ばれるものがあり、さらにクリティカルアプリケーション分野がある。人命に関するものがこれにあたる。

(3) 技術

マンマシンインタフェースを例にとると日本語、グラフィックス、イメージなど入出力機器と、エンドユーザ言語やかな漢字変換辞書などの狭義のソフトウェアの技術範囲と、価格低下をどう見るかという時点の設定がある。

(4) 価値感

価値感の変化や文化的要素をどの範囲まで考慮するか。

以上のように「ソフトウェアの使いやすさ」については、それに関心がよせられる背景も評価要因も多面的であり、検討の対象範囲も広く評価尺度も一律ではないといえる。

しかし使いやすさが、ソフトウェアの重要な評価要因であることは疑う余地のないところである。みんなで知恵を出し合い、経験を生かし合って取り組んでいきたいテーマである。本討論会がソフトウェアの使いやすさの推進に大きく寄与することを期待している。

(司会 福田 善一)

ユーザの立場から

伊藤 正雄

1. はじめに

近年、ハードウェア、ソフトウェア技術の飛躍的な進歩に伴い、コンピュータ利用は急激に拡大しつつあり、その利用形態の標準化、ソフトウェアのパッケージ化等も急速に進みつつある。

我々がコンピュータに業務を実行させることは、当然ソフトウェアを利用することと同義であり、ソフトウェアの使いやすさは、ユーザの立場から重要な関心事とならざるをえない。

ここでは、鉄鋼業におけるコンピュータ利用を例にユーザの立場からのソフトウェアの使いやすさについて述べることにする。

2. 鉄鋼業のコンピュータシステムについて

鉄鋼業におけるコンピュータの利用は、歴史的に見ても早い時期に始まり、製造現場である製鉄所を中心に発展してきた。現在では、ビジネス・コンピュータは生産を中心に諸企画管理業務等ほとんどすべての分野をカバーし、生産設備の自動運転、プロセス制御をサポートするプロセス・コンピュータと階層的なネットワークを構成している。

鉄鋼業では、一品ごとの受注生産方式の生産管理を行っており、製造途中で起る異常・例外処理をコンピュータにさせるために、アプリケーション・ソフトウェアは膨大なものとなっている。

3. 使いやすいソフトウェアのイメージ

鉄鋼業においては、20年余りのコンピュータ利用の歴史を経て、今日、日常業務を遂行する上でコンピュータは欠くことのできない重要な管理の道具となっている。

この道具という見方からすると“ソフトウェアの使いやすさ”は普通の機械の使いやすさと同様に考えることができる。

元来、道具というものは、ユーザにとって、内容とか構造を知り尽くさなくても使えるものでなければならない。佐貫亦男教授が「道具の再発見」で述べられているように、「一目でわかること」が必要であり、「動くべきものは動き、動くべきでないものは動かないこと」、「軽く、快く、歯切れよく、機能的に動くこと」、「愛着がもてること」等が要請される。更に付け加えるならば、コストが適正であることも必要であろう。

一方、ソフトウェアを利用するユーザにも、いろい

ろな立場、様々な技術レベルの人達がいる。以下では、通常企業内でコンピュータと接する代表的な二つの立場、すなわちエンドユーザといわれる業務の実行箇所の立場と、DP (Data Processing) あるいは情報システム部門と呼ばれるシステムの設計、製作、運用にあたる人の立場から問題を整理してみたい。

4. エンドユーザから見た使いやすさ

実務の第一線にいるエンドユーザは、コンピュータの専門家ではない。したがって、彼等が直接利用するソフトウェアは、道具を意識せずに機能を果せること、いわゆる fool proof であることが要請される。

これには、二つのレベルがあると思われる。

① 初級者向：徹底した fool proof

単純化された機能、操作性への要求である。ただし高級な機能までは必要としない。

② 中級者向：エンドユーザでも、自分でシステムを手造りしたい人への対応として適当なユーザ向きの言語が必要である。

いずれのレベルに対しても、人間的な、特に日本人向きのマンマシンインタフェースの設定が要請される。また、このような機能に対応するハードウェア資源負荷が適当でなければ実用にならない。また解りやすい利用手引書や端末画面を用いてのガイド機能も必要である。

また、信頼性も重要である。エラーを起さない、タフなシステムであるとともに、ユーザのミスに対するリカバリ機能をもつことが望まれる。ユーザ原因でトラブルが起っても、適切なメッセージで原因と処置を表示する程度の配慮が望まれる。

5. 情報システム部門にとっての使いやすさ

各種の半製品のソフトウェア・モジュールを組み合せ、また追加、改造し、エンドユーザのニーズを満たすソフトウェアを設計・製作する人の立場から次の点があげられる。

(1) 選択、組合せ使用に際して、機能、機構仕様やインタフェース仕様が明確であること。

(2) アプリケーション・システム設計段階で性能(ハードウェア資源負荷等)が予見できること。

(3) 信頼性、バグがなくトラブルを起さないこと。

(4) 柔軟性、特に環境の制約がなく、ユーザが自由に選択・組合せできること。

(5) コストが適正であること。

6. 具体的な問題点と要望

若干の例を示しながら問題点と要望を述べる。

(1) パラメータの設定が煩瑣

- ① オペレーティングシステムのシステム・ジェネレーション
- ② ジョブ制御言語の諸パラメータ
これは機能上の指示だけで済まないのか。
- ③ 直接アクセス記憶装置上のブロックサイズ等
レコードの論理的な定義だけで済まないか。

ハードウェアから独立して、論理定義、機能定義だけで使えるソフトウェアが強く望まれる。

(2) パッケージ間のインタフェース

現在、数多くのソフトウェア・パッケージが流通しているが、パッケージ間のインタフェースが統一されていないために、組み合わせ使用することが困難である。たとえば、同一メーカーであっても汎用データベース管理・通信制御システム、タイムシェアリングシステム、専用パッケージ間の相互乗入れが困難な場合も多い。

(3) モジュール化

モジュール構造になっていないソフトウェアが多く、アプリケーション機能と対応したモジュール分割、モジュール間インタフェースの整備が望まれる。

(4) 利用手引書 (マニュアル)

ユーザサイドに立って作られたものが少なく、一般の利用者には、難解であり、索引等の充実が望まれる。

7. 業界全体への要望

以上述べた諸点についての各々の改善もさることながら、業界全体で取り上げてほしい問題として次の各項目がある。

(1) 機能、性能の記述方式の統一

このためには、用語の統一と機能、性能表示方式の研究が必要である。

(2) 基本ソフトウェアのユーザ・インタフェースの標準化

オペレーティングシステム、データベース管理・通信制御システム等の基本ソフトウェアはなお発展中であり、ユーザ・インタフェース固定化には全く問題がないわけではないが、各種ソフトウェアを組合せ利用できるよう、可能なものから標準化を進めて頂きたい。

(3) 購入契約条件の標準化

ソフトウェア流通促進のため、品質保証、トラブルへの対処等に関する契約条件の整備、標準化が必要である。

8. おわりに

以上、ユーザの立場からソフトウェアの使いやすさ

について現状の問題点とメーカー側への要望を述べてきた。ハードウェアの発展が着実に進む中で、今後の情報化の鍵は、まさに使いやすいソフトウェアの開発にかかっていると見て過言ではないと思われる。この問題について、メーカー、ソフトウェアハウス、ユーザの三者が緊密な関係を適正な分担のもとで積極的に取り組むことが望まれる。

使いやすさを実現するための生産技術

片岡 雅憲

「ソフトウェア」を「計算機利用技術」と訳すことがあるが、これはハードウェアをいかに上手に利用するかとの実体に乏しいハウトゥ (How-to) 的なものを類推させるがゆえに危険な訳語であるといわざるをえない。ハードウェアの進歩とともに、機能の高度化と量的拡大を遂げた現在においては、ソフトウェアはハードウェアと全く同様の具体性を持った工業製品として、巨大な機械としてとらえるべきである。文字どおり、ソフトウェアは物 (ware) であり、利用する技術ではなく、ソフトウェア自身が利用され、使用される物でなければならない。

ソフトウェアをこのように物として、工業製品としてとらえたときに、その「使いやすさ」が問題となる。そして、メーカーの立場からは、この「使いやすさ」を実現するための生産技術が重要となる。以下では、このような観点から、ソフトウェアの使いやすさと生産技術との関係を考えてみることにしたい。

1. 使いやすさと品質

「使いやすさ」を論ずる場合には、当然、「使用者 (ユーザ)」がだれであるかを明らかにしなければならない。一般に、ソフトウェアの使用としては多様な種類の、多様なレベルの人々が考えられる。たとえば、入力データを作成する人、端末オペレータ、センタオペレータ、センタ管理者、出力データの利用者、当該ソフトウェアの導入を推進する情報処理技術者、その上長である合理化推進担当管理者、等々、色々な人々をあげられる。「使いやすさ」はこれらのすべての人にとっての「使いやすさ」でなくてはならない。

また、このように考えたとき、「使いやすさ」は、コンピュータを直接的に操作する場合の操作性だけでなく、性能、信頼性、拡張性等々の多面的な特性から構成されていることがわかる。したがって、「使いやすさ」とは、より一般的には「使用者の満足度」ととらえることができる。そして、これは近代的品質管理

における「品質」の定義にはかならない。すなわち、

ソフトウェアの使いやすさ＝ソフトウェアの品質との定式化が成り立つ。したがって、「使いやすさ」の問題を議論するとき、品質管理の諸概念が活用できる。

2. ソフトウェアの品質の概念

使いやすいソフトウェアを実現するためには、

設計品質：ユーザ要求に仕様が適合していること

プログラム品質：仕様どおりにプログラムが動作すること

の両者を満足させる必要がある¹⁾。とすると、プログラム品質の良否だけをもって、ソフトウェアの品質が云々されることがあるが、誤解が生じぬように注意する必要がある。「使いやすさ」というユーザの観点からの品質概念に基づけば、優れたプログラム品質は、優れたソフトウェア品質を実現するための必要条件であっても、十分条件ではない。

一般に、ソフトウェアは、使用環境の変化や、ユーザ要求の高度化に対応するために、仕様の変更や拡張を余儀なくされる。この点に関して、ロンドン大学の M. Lehman は次のように述べている。

「ソフトウェアは、それが生きている限り、絶え間なく変更される。変更されなくなるのは、それが死んだ（使われなくなった）ときである。」

このように、ソフトウェアについてもライフサイクルが存在し、そして、このライフサイクルは①初期運用、②保守・拡張、③移行・接続の3段階から構成される。ソフトウェアの品質特性も、このライフサイクルの観点から定義する必要があり、図-1に示すようにライフサイクルの各段階と対応づけられる。

品質特性（真の品質特性）は、ユーザにとっての使いやすさ、ユーザの満足度を表す指標でなくてはなら

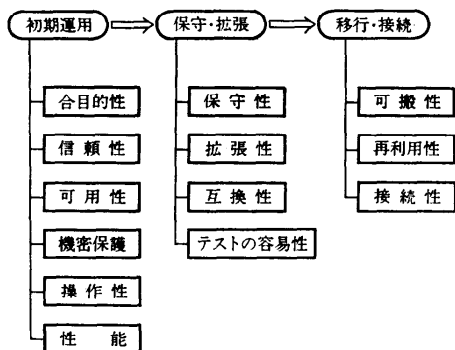


図-1 ソフトウェアライフサイクルと品質特性との対応

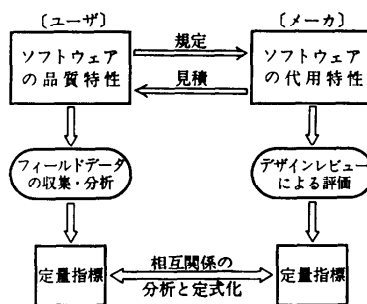


図-2 品質特性と代用特性の相互関係定式化の方法論

ない。したがって、品質特性は、ユーザの使用環境下で測定が可能であって、ソフトウェアの生産工程で測定することは困難である。生産工程では、ユーザの使用環境を擬似することによって品質特性を測定するか、または、これに代る代用特性（代用品質特性）を測定する必要がある²⁾。たとえば、保守性（不良原因を容易に解析でき、また、修正できること）という品質特性は出荷前に測定することはできないから、代わりに、機能構造やモジュール構造の良さ等の代用特性で管理することになる。したがって、ここで大切なことは、品質特性と代用特性の関係を統計的に分析して、定式化していくことである（図-2 参照）。

3. ソフトウェアの構造と生産技術

ソフトウェアの生産工程は、ユーザ要求というきわめて大づかみで抽象的なものを、段階的に詳細化しながら、最終的には具体的なプログラムのコーディングステップに変換していく過程である。これを構造的にとらえると、論理的構造から物理的構造への変換と、階層構造化の二つの作業を同時に混在させながら進めていることになる。そして、このような構造的なアプローチこそ、ソフトウェアの使いやすさを実現するための生産技術に欠かすことができない。これは、次の理由によるものである。

(1) 品質特性と代用特性との関係の定式化のためには、構造的アプローチが必須となる。

(2) 系統的設計技法及びテスト技法は、構造の標準化と密接に結びついている。

(3) ソフトウェア開発支援ツールの効果及び性能は対象とするソフトウェアの構造の標準化度合に強く依存している。

4. まとめ

「ソフトウェアの使いやすさ」を考えるときに、まず第一になすべきは、品質管理の立場から「使いやす

さ」を定義することである。そして、その根底にあるものは「徹底したユーザニーズ指向」である。次に、大切なことは、このユーザの観点から規定された特性を、ソフトウェアの生産工程へ、系統的に展開していくことである。そして、この系統的展開は、ソフトウェアの構造の標準化なしには実現できない。すなわち、「ソフトウェアの使いやすさ」を実現するための生産技術は、単独では存在しえず、品質管理技術と、構造化技術との有機的結合によってのみ、実現可能である。

参考文献

- 1) 狩野：製造業・建設業における品質管理の考え方および手法のソフトウェア産業への活用，第1回ソフトウェア生産における品質管理シンポジウム発表報告集（昭和56-7）。
- 2) 石川：日本の品質管理，日科技連出版社（昭和56-11）。

今われわれにできること

木村 泉

ソフトウェアは使いやすくてはならぬ、とは昔からいわれてきたことだが、10年20年前にその主張をまじめに受けとめた人は、むしろ少なかったのではないか？ 貴重な計算機資源を使いやすさのために浪費するなどのもつてのほか、利用者を甘やかすにも程があると、口に出していわぬまでも内心思っていた人は、ずいぶん多かったはずである。近来、この問題が広く各方面の関心を集め、正面から議論されるようになりはじめていることは、まことによろこばしい。

ソフトウェアの使いやすさとは、計算機と利用者から成る人間機械系をひとまとまりのものとして見たとき、系に所定の目的を達成させるために必要とされる利用者側の苦勞が少ないことである、といつてよからう。しかるに利用者の苦勞を少なくしようと思えば計算機側で必要とされる手当ての総量はとかく急増するものである。かつてはハードウェアが高価であったし、ソフトウェア構成技術もまた未熟であったから、利用者側に多少の苦勞を強いてでも計算機側の手当てを省こうとすることは、経済性の原則からみて至って自然なことであった。計算資源が当時とは比べものにならないほど豊富になってきた今、ソフトウェアの使いやすさは、論じ得るようになったのみならず、論じざるを得ない状況になってきたといえる。

さてそうなった今、われわれはソフトウェアの使い

やすさについて、どんなことができるだろうか？ 次の四つが考えられる。

1. (逸話的) これまでのソフトウェアがどんなに使いにくかったか、その事例を集めて人々の認識をうながすこと。
2. (政治的) ソフトウェアの使いやすさが重大な問題であることを、大家に宣明していただくこと。
3. (実践的) 使いやすいソフトウェアの作りかたについて、経験、ノーハウを蓄積すること。
4. (科学的) ソフトウェアの使いやすさについて自然科学的客観事実を確立すること。

このうち1はずいぶん古からおこなわれてきたことであるが、計算機が高価であった間、とかくいまひとつの迫力を欠き、おこなわれるにしても多少とも娯楽的な読み物の中でおこなわれることを常とした。いまやこういう方向も十分意義をもち得る時代がきたわけであり、まじめな態度でいっそう推し進める必要があると思う。

それにしても1のような方向は、とかく「ごまめの歯ぎしり」におちいりやすい。ソフトウェアの使いやすさの問題を正面から論ずべき時代がきたことは、ソフトウェア作りの現場にある者にとっては明らかであるとしても、少し離れたところにいる人々には必ずしもそうでない可能性がある。老大家に大きな声でいていただいて、世の人々が「ほほう、そんなものか。」と思ってくれるように仕向けることは大いに意義がある。これが2である。

一方使いにくい使いにくいといいたてることもたいせつだが、どうやると使いよいかをはっきりさせることはもっとたいせつである。これが3である。たとえばUNIXオペレーティングシステム¹⁾には、そういう使いやすさの範例を示したという意味で非常に大きな意義があったと思う。UNIXは利用者が口を揃えてほめちぎるというふしぎな性質をもったソフトウェアである。実際、使ってみると身体が楽で、仕事がたいへんはかどる。こういうソフトウェアが確実に作れるようになるためには、ノーハウの蓄積がたいせつである。

最後に4は、困難な方法であるが、少しずつ成果が出はじめており、いずれは使いやすさの問題についてもある程度の理論化工業化ができるのでは、と期待させるほどのものを持っている。この方向での一つの輝かしい成果は、Card-Moran-Newellの“keystroke-level model”の仕事²⁾に見られる。会話型システム

でよく慣れた定形作業をおこなう場合の作業時間という、ごく小さい事項に話を限定してはいるものの、はっきりした、実用的にも意義のある結果を出している点、高く評価できる。

さて3, 4のようなことをやって行くためには、使いやすさという特性の多元性を認識することがたいせつである。よくいわれることだが、使いはじめの使いやすさと慣れたあとの使いやすさはちがう。小さいことをちょっとやるのに使いやすいか、大きいことをしつづくと慎重にとりおこなうのに使いやすいかも大きなちがいである。

たとえば、上で引き合いに出した UNIX は使いやすいことで有名だが、その使いやすさはたぶんズブの素人のものではない。巨大な定形業務を日常的にこなさねばならぬデータ処理関係者にとっての使いやすさでもなさそうである。その辺をはっきりさせなければきっちりした議論はできない。

とはいうものの、もう一つもっと大きな考慮点があるように思う。それはわれわれが過去から引きずってきた、常識という名の惰性である。昔身についた貧乏性が抜けないため、われわれはしばしば自明なことを見落す。そういうところから脱脚するためには、分析的に考えてみる以前にまず、利用者に「このソフトウェアは使いやすいですか？」ときいてみるストレートさがたいせつなのではないか？ もちろん「利用者」とはだれなのかは大きな問題だし、中には無責任な答をする人もいるだろうが、ともかく皆が揃って使いにくいというものはどこかおかしいにちがいない。それを「計算機というものはそういうものだから」といって静めようとするのは、今となっては時代遅れである。たぶん現時点でもっともたいせつなのは、この点の認識だと思う。

最後に、パネル討論の席上他の講演者から提示された下記の二つの興味深いテーゼについて、この場を借りて筆者なりの見解を示しておきたい。

テーゼ 1, ソフトウェアの使いやすさとは、それがひっくり返らないことである。たしかに始終ひっくり返るソフトウェアは使いやすいソフトウェアとはいえない。だが、ひっくり返りさえしなければ使いやすい

い、ともいえない。マニュアルに書いてある通りに使えばたしかに書いてある通りに動くソフトウェアでも、そのマニュアルに書いてあることが利用者の思考の自然な流れを一々せきとめるようなものであったとすれば、やはり使いにくいソフトウェアであるといわなければならない。たとえば会話型システムにおいて、同じようなことをするための指令の形式が(たとえばコマの数やパラメータの入力形式について)少しづつちがっており、そこをまちがえると頑として動かないようになっていたとすれば、そういうシステムの利用者はとかく神経を消耗し、高い生産性を達成することができない。

テーゼ 2, 子供が自動車の運転をすることは許されないと同様、ソフトウェアも、利用者に自動車学校で要求される程度の訓練を要求してよい。自動車の運転に免許が必要なのは自動車というものがまだ理想的水準まで発展しつくしていないからである。前方に歩行者がいたら自動的に停止するとか、本線上の車の流れを見て側線から正しいタイミングでとび込むとかは自動車の方で自動的にやってくれたとしても決してわるくない。技術の現状ではなかなかそうも行かないので、やむを得ず運転者に厳しい訓練を課すことで補いをつけているにすぎない。しかるに世のソフトウェアはとかくその点の反省を欠き、設計のくふう次第どうにもなるところで利用者の習練を不当に要求していることが多い。習練を要求するにしても、計算機側の都合と利用者側の苦勞の間の微妙なバランスの上に立って要求するのだからなければならない。自動車学校なみの習練を要求することはほとんどすべてのソフトウェアについて、過大であると考える。

参 考 文 献

- 1) Special Issue on UNIX Time-Sharing System, Bell Syst. Tech. J., Vol. 57, No. 6, Part 2, pp. 1897-2312 (July-August 1978).
- 2) Card, S.K., Moran, T.P. and Newell, A.: The Keystroke-level model for user performance time with interactive systems, Comm. ACM, Vol. 23, No. 7, pp. 396-410 (July 1980).