報　告

パネル討論会

ソフトウェアの評価

創立 15 周年記念大会　報告

日時　昭和 49 年 12 月 6 日 (金) 13:30～16:00
場所　国立京都国際会館 D 会場

パネリスト

(司会)(報告者)　大野　豊(京都大学工学部)
山本欣子(日本情報処理開発センター)
高橋延匡(日立製作所)
高橋　理(東北大学大型計算機センタ)
石田晴久(東京大学大型計算機センタ)
石崎純夫(富士銀行)

この報告は本学会創立 15 周年記念大会で行われたパネル討論会「ソフトウエアの評価」の記録として当日のパネリスト達によってまとめられたものである.

なおパネリストのうち高橋　理氏には,当日病気で出席できなかった戸田　厳氏(電電公社横須賀通研) に代って急に出て頂いたものである. さて以下は当日の各パネリストの発言を各自まとめて頂いたものである. 会場では, 多くの方々から意見や質問がなされたが, それらをすべて問答形式で紹介する紙数がないので, それらの討論の内容も含めるよう各パネリストにお願いした. したがって当日の質疑応答の雰囲気が必ずしも忠実に反映されていない点はご了承頂きたい. またこれらの討論に参加して下さった方々のご了承も併せてお願いしたい. (大野　記)

ソフトウェアの評価について

大野　豊

計算機技術はかなり高度に発展したが, ソフトウェアについては, 普通の文章を書くのと同じように, 人間がこつこつと手で書くこと, すなわち手づくりが基本的にはさけられない. このため, ソフトウェアは科学でなく craft であるとさえいわれている[1]. この手づくりであるということから, ソフトウェアが大規模あるいは複雑になると, 多くの問題が生ずる. たとえば, テストやデバッグに手間がかかって, 結局, ソフトウェアの開発に多くの時間, 人工, 費用がかかるようになる, というのもその代表的な問題点である.

このようなソフトウェアにもっと科学的工学的方法を適用して, これらの問題点を改善し, よいソフトウェアにしようとするのがソフトウェア工学の目的であろうが, ここで,"よいソフトウェア" とは何かということが基礎的問題として浮び上ってくる. もし,"よいソフトウェア" の基準が明確に示されるならば, 与えられたソフトウェアの評価が可能となり, したがって, "よいソフトウェア"を作る方法論もできてくるであろう.

"よいソフトウェア" とは何か, どんな条件が満されればよいか. この問に答えるのは容易ではない. ソフトウェアを評価する項目は, ソフトウェアの種類, 目的, 環境, 立場 (利用者か供給者か) その他, 多くの因子によって異なってくるであろう. 本パネル討論会が開かれたのは, このようなソフトウェアの評価に関係する因子をすべて洗い出し, 評価項目として整理し, さらにそれらの項目についての評価の方法や技術を提示することに期待があったためであろうが, 技術の現

状のほか本討論会での準備時間の関係もあって，この期待に必ずしも沿いえないのは残念である．

ソフトウェアの評価は多次元的なものであり，評価の方法も多様であり，しかも充分に確立されるところまでには至っていない．しかし，非常に基本的なレベルでのソフトウェアの評価の因子は示されている．MIT の J. B. Dennis 氏はソフトウェア設計者の目標

左から大野，山本，高橋(延)の各氏

は次の4つの性質によって表現されるとしている[2]．

(1) function
(2) correctness
(3) performance
(4) reliability

これらについて少し説明を加えれば，まず最初の **function**（機能）は，ソフトウェアの入力に対する出力の望みの対応をいう．たとえば，FORTRAN コンパイラは FORTRAN でかかれたソースプログラムを機械語のオブジェクトプログラムに変換するのがその主機能である．しかし，ここでいう機能はこのようなことだけではなく，たとえば，操作上，取扱上の諸機能などは副次的なものではあろうが，重要な機能である．

correctness（正確さ）というのは，ソフトウェアの目的とすることが，その言語で正しく記述されていることをいう．この言語がそのホストシステム上で正しく動作するならば，その言語で正確に表現されているソフトウェアの動作の正しさは保証される．

performance（性能）は，そのソフトウェアのホストシステムのリソースが，そのソフトウェアの目的をみたすのに，どの位有効に使われるか，を示すものである．したがって，CPU 処理時間，メモリ使用量，スループットなどをはじめ多くの項目がこの中で考えられる．ソフトウェアの評価というと，まずこの性能が問題にされ，多くの研究発表もされている．

reliability（信頼性）は，そのソフトウェアが，計算機システムの要素の障害にもかかわらず，その機能を正しくいかに遂行しうるか，を示すものである．ソフトウェアは一度つくられたらこわれない．普通によく“ソフトウェアの障害”といわれるものは，したがって，正確さに関していうことが多い．フェイル・ソフトという言葉は，ここでいう信頼性を一応別のいいあらわしをしたものと思われる．

以上の4つの性質を，さらに詳細に分析すれば，ソフトウェアを評価する諸因子を抽出することができるであろう．本討論会では，パネリストの方々が，それぞれ異なった立場からソフトウェアの評価について論じておられるが，そこにのべられている種々の評価項目は，上記の4因子から派生してくるものといえるように思われる．

本討論会では，評価の項目と方法を整理して論ずるより，むしろそれぞれの異なった立場から，多次元的に評価の問題点を示し，また評価技術の現状をも示すことによって，この方面への大方の関心と参加をもとめようとしたわけである．

計画では，大学，ユーザ，メーカ，その中間的なソフトウェア発注者といった立場の方々を予定したのであるが，前述のようにメンバの一部の突然の変更があり，多少計画とは異なったものとなった．

また，短い時間の討論でもあるので，あらゆる問題をすべて論ずるより，問題を限定した方がよいであろうということで，ソフトウェアとして，オペレーティング・システムを中心とし，それとの関連でアプリケーションのソフトウェアも考えよう．ということであったが，実際の討論では，言語そのものに対しても，多少の議論がなされたり，評価というより，よいソフトウェアを生み出す方法論，たとえば構造化プログラミングについても議論が行われた．これらのことは，ソフトウェアの評価と密接に関係したことなので，当然関連の議論として行なうことは，さけられないことといえよう．

(註) 1) E. Yourdom: “Design of On-line Computer Systems”, Prentice-Hall, Inc., Engle wood Cliffs, New Jersey, U. S. A, 1972 (大野，落合，監訳)
2) F. L. Bauer, “Advanced Course on Software Engineering” Springer-Verlag, Berlin. Heidelberg. New York, 1973.

評価システムについて

山本欣子

ソフトウェアの評価を行なうにあたっては，評価の項目，規準，手法などが問題となる．

1972年の某月某日，数名の学識経験者が集まって討議したアプリケーション・プログラムの評価項目というものを参考までに挙げてみよう．

①カタログ性能通りの正しい動作 ②フェイル・ソフトの機能の完備 ③精度 ④処理時間 ⑤メモリ使用効率 ⑥コンピュータ使用時のコスト効率 ⑦ソフトウェア自体の妥当な価格 ⑧メインテナンス費用 ⑨アフターサービス性 ⑩利用時の学習が容易 ⑪操作性 ⑫誤使用，誤操作の検出性 ⑬他のソフトウェアとのインタフェースのとり易さ ⑭ドキュメントの完備 ⑮修正，変更，再構成が容易 ⑯技術的に長期の使用に耐え得る ⑰オペレーションが容易 ⑱ある程度のマシン・インディペイデイト性

これらの項目をあげるに際して次のような点が前提となる．

① アプリケーション・プログラムは適用目的により評価のポイントが変わるのが当然で，上記の各項目は比較的共通した基本的なものに過ぎない．

② それぞれのソフトウェアのカタログ性能あるいは機能仕様は，むしろそれを使用するユーザが個々に独自の目的に合わせて自分で評価すべきである．一般的評価はあり得ない．

③ 業種別あるいは分類別のソフトウェアは，上記基本項目に加えて夫々の専門分野で個々に評価項目あるいは基準が作られるべきである．

④ 現在なすべき事は，代表的なアプリケーションプログラムをモニタリングやベンチマーク等の可能な手段も利用して上記基本項目につき調査測定し，評価基準作成の参考となるデータを収集する事である．

さて，次に具体的な評価を行なう道具としての評価システムというものにつき若干考えてみよう．

ハードウェア，ソフトウェアを総合したシステム評価のためには，一般にベンチマーク，シンセティックプログラム，シミュレーション，ハードウェアモニタリング，ソフトウェアモニタリング，解析的モデリング等の手段あるいはシステムが使われている事はすでに周知のことである．

これらの一部を含め，特にユーザプログラムあるいはアプリケーション・プログラムを対象としたものも若干ある．ソフトウェアの評価項目には定性的なものと定量的なものとがあるが，コンピュータを利用した評価システムという面からは当然定量的なものが中心となる．またある程度は定性的な要素を何等かの形で定量化するという傾向もみられる．しかしいずれにしろ数多くの評価項目のうち大なり小なり評価手段のシステム化が可能だとされている部分は現在まだきわめて限られているが，若干の例を示そう．

1. 仕様規準に対する検定

ソフトウェアの仕様には，ある種の規準が設けられる場合がある．その製品がその規準に合格しているか否かを検査するもので，現在のところ米海軍の COBOL コンパイラ検定システム註)が有名である．

(注) 1. "米国海軍 COBOL 検定システムの検討" 昭和 47 年 (社)日本電子工業振興協会報告書
2. "新しい COBOL コンパイラ検定システム" 植村：電総研，第 16 回プログラミング・シンポジウム報告書

これは米国のスタンダードである ANS COBOL の言語仕様の検定を行うことを目的としたもので，その COBOL コンパイラが

左から高橋(理)，石崎，石田の各氏

(1) ANS 規格によるすべての言語要素を含むか
(2) それらの機能が，規準に合った解釈をしているか
(3) 正しく翻訳され，結果が正しいか

等を検定する．したがってコンパイラのエラーや効率を診断するものではない． COBOL 全般にわたり約 3,000 種のテスト項目からなり逐次改良されている．米国ではかなり広く使用されているという．

2. 効率評価

ユーザプログラムを対象としたものは FORTRAN または COBOL のプログラムに関するものが殆んどである．これらはユーザプログラムを実行時にモニタリングし，①各ステートメントの実行回数 ②それに要する時間 ③全体に対する％ ④IF などの分岐の頻度等，どこを改良すれば大きく効率が上がるかを示す効率評価のための情報を出力する．これらのシステムの効果としてはパフォーマンス・バグの発見による処理時間およびメモリ・エリアの節約にあるわけだが，やや具体的な効果としては ①頻度高くコールされるプログラムを1モジュールにまとめる ②テーブル内の各エレメントの参照頻度によるテーブル内再配置 ③データ属性の変更による参照時の変換の除去 ④パラグラフ単位の遷移確率にもとづくプログラム再構成等の例があげられる．なおシステムによっては評価情報を出力するだけでなく，最適化したソースプログラ

ムを自動的に作成するものもある.

3. 品質保証システム(註)

自動化されたソフトウェアの品質保証システムは次のような機能を持つ.

①テスト効果の定量化　②テスト・プロセスの自動化　③テストケースの自動的な改良

具体的方法の例としては，テストデータをある法則にしたがって自動的に作り出すテスト・データ・ジェネレータの機能と，そのデータによるプログラム実行時の各ステートメントのトレース機能を組み合わせ，実行されたステートメントと実行されなかったステートメントの比率から，テスト効果の定量化をおこなう．すべてのステートメントの実行が確認され，それらのテスト・データによる結果がすべて正常であれば（プログラムの正確さという意味における），ある程度の品質の保障が可能である.

以上はごく一部の例に過ぎないが，いずれにしろソフトウェア評価の自動化はまだ今後の問題と言えよう．しかし，この 10 数年来，ソフトウェアの開発技術は，ハードウェア自体あるいはコンピュータそのものの普及という周辺環境の著しい進展に比して，唯一つ取り残された感がある.

この原因の一端は，ともかくも目先の仕事に追われ質よりも量のつぎ込みに専心せざるを得なかったという事情があるが，今後の課題はソフトウェアを単に生産するだけではなく，**質のよい**ソフトウェアを生産することであろう．そのための手段として，まずよいソフトウェアとは何かという評価の規準を確立し，モチ屋はモチ屋らしく評価の道具としてコンピュータ自体を充分に活用すべきであろうと思う.

(注) "ソフトウェアの品質保証の自動化について" J. R. Brown 他　鳥居宏次訳「プログラム・テスト法」より

オペレーティング・システムの評価

高橋延匡

ソフトウェアの評価に関するパネル討論会の企画がとりあげられた背景の一つは「計算機システムのコスト・パフォーマンスは年々向上しているが，ユーザにとって本当に使い易く，便利になっているのかどうか」という素朴な問題の提起にあったと思われる.

ここで，私がとりあげたいのは，HITAC 5020 モニタ，5020 TSS[1)2)]の開発経験と 5020 TSS の評価[3)4)]に取り組んで来た経験を通して，現状でのオペレーティング・システム (ÕS) の評価の問題点を明らかにすることである.

ÕS の設計という見地から考えてみると，第二世代計算機システムの ÕS に関する設計目標は主として，**連続処理**（バッチ処理）によるスループットの向上に置かれていた．また，計算機のアーキテクチャも実メモリ系の方式であり，設計時点におけるシステム性能の見通しと開発完了後の性能実測値とは大差はなかった．ところが，第三世代計算機システムにおいては，ハードウェア技術の進歩にともない，使い方の多様化に対処するため，ÕS の汎用化が進んだ．現実の第三世代計算機システムの ÕS は，技術的にはマルチ・プログラミングを基調にしているが，設計目標のちがいにより，二つの大きな流れ（力点のちがい）がある．一方はバッチ中心の汎用 ÕS であり，スループットで代表される性能を第一の目標とする．他方は汎用 TSS であり，マン・マシン性（応答性能）を第一の目標とするものである．**表-1** にその主な特徴を示す.

表-1 汎用 TSS と汎用 OS との設計目標の比較

リモート・アクセス中心の汎用TSS	バッチ中心の汎用 ÕS
レスポンス・タイムが目標	スループットが目標
オンライン・アクセス中心	バッチ中心
バックグラウンド・バッチ	タイムシェアリング・オプション
オンライン端末	リモート・ジョブ・エントリー
便利なコマンド（端末操作）言語	バッチのジョブ制御言語と共通
ファイル処理はメモリとしてアクセス	ファイル処理は入出力装置としてアクセス
プログラムのリスタートが容易 (Quit)	check-point として計画的に
dynamic link が中心	static link が中心
protection の強化が必要（特にプライバシー）	protection，特にシステムの安全性が重要
問題向言語 (Basic, APL, …) が重点	汎用言語(FORTRAN, COBOL)が重点
ユーティリティの提供	コンピューティング・パワーの提供

したがって，表-1 から明らかなように，スループットに重点を置いて設計されたシステムに対し，後者の特徴であるマン・マシン性の良さが不足している点を強調することとか，反対に後者に重点を置いて設計されたシステムに対し，前者と同様なバッチのスループットが得られない事などの指摘は設計目標の設定上の問題から来たものである．もちろん，両方を十分に満たす方向への努力が各メーカーでなされていると思われるが．これは計算機のコストの上昇につながる可能性もあるが，この方向に進んでいることは事実である.

以上は ÕS をある機能的な面から評価し，選択する場合の基本的な考え方についてである.

一方，性能の問題であるが，昭和 43 年に仮想メモリ方式の汎用 TSS として 5020 TSS を実験的に開発して使用してみると，仮想メモリ容量に対して実メモリ容量が余りにも少なかったため，二次メモリ（磁気ドラム）と主メモリとの間の情報の交換 (Swapping) による ÕS のオーバヘッドが使用条件によって非常に増加する場合があることが判明した．そのことから，仮想メモリ系の ÕS の性能評価の必要性を痛感し，ソフトウェア・モニタによる計測，プログラムの動作解析，スワッピング・アルゴリズムの評価等を行なった．しかし，それ等の結果からは，ÕS の改良指針は得られたが，たとえば，「主メモリを何キロ語増設したらレスポンス・タイムはどの位はやくなるか」あるいは，「何台の端末が実用可能か」など基本的な質問に対し明確な解答ができないので，ÕS のシミュレーション・モデルによるシミュレーション評価に頼らざるを得ないことになった．

シミュレーションによる ÕS 評価の問題点は，いかに信頼出来る精度の良い結果を得るかという事である．そのために，ÕS のモデル化をかなり詳細に，かつ，大胆に行なう必要がある．また，モデル化にあたっては，

（1） 評価対象に関連する部分は詳細なモデル化が必要．

（2） 自分が開発に関連した ÕS でないとモデル化に時間がかかる．

（3） ÕS のバージョン・アップに追従する必要がある．

を十分に計算に入れておく必要がある．

以上，ÕS の性能評価は，計算機システムの複雑化傾向に対処するためにも，手法，評価の道具の開発，評価規準の確立が重要な課題となる．

評価とは物の値をはかり定めることである．しかし，現在までのところ，ÕS の総合評価のメジャーは確立されていない．メジャーを確立することは最大の課題であるが，一朝一夕にできることではない．しかし，「ÕS の評価」の目的は，，「新しい計算機システムを設計するために必要とする機能，性能について，具体的な目標設定に役立たせること」である．したがって，地道に，測定可能な種々のデータに関し，現実のシステムの計測，解析を通じ，改良に生かしながら，設計技術の蓄積をはかることが，現時点での課題と考える．

参 考 文 献

1) 嶋田，他："HITAC 5020 TSS の概要"，他 12 件，昭 43 電気四学会連合大会論文集，分冊 31，情報処理（1），pp 2956～2968

2) 本林，他："2 次元番地付方式による HITAC 5020 TSS の特徴"，情報処理，Vol. 9, No. 6, pp. 317～325 (1968)

3) 高橋，益田："HITAC 5020 TSS の解析と評価（プログラムの動作解析）" ÕS シンポジュウム報告集，情報処理学会 70 年 9 月

4) 益田，他："ページング・マシンにおけるスワッピング・アルゴリズムの比較とプログラムの動作解析"，情報処理，Vol. 13, No. 6, pp. 81～88 (1972)．

ソフトウェア評価の考え方

高 橋 理

私はまずソフトウェアとハードウェアを独立に評価する問題が成り立つかどうかという観点からハードウェアとの関連におけるシステムの評価とどう違うかをまずお訊ねしたい．仮に抽象的にシステム記述言語を考え，ソフトウェアの機能という点では「何ができるか」というカタログ的なことは評価できても数量的性能の評価が果してできるか？ コンピュータの特徴は現実的な時間内で行えるという条件付でいえると思う．ソフトウェアの評価法が確立したかどうかの評価の目安として，例えば美辞麗句の並んだカタログを正しく読むことの成否で測ってはどうか？ 試みとしては，メーカ名をかくしてソフトウェアに関するカタログの総合評価をうまくやれるように基準化することが考えられる．

次にコンパイラなどで評価によく出てくるものに機能面として言語水準の高さとハードウェアの潜在能力を活かすオブジェクトを作り出すことが挙げられ，後者が昂じると言語に対する文法や解釈，常識にふさわしくない処理も出て来る恐れがある．然し，大学のユーザ達にはこれに劣らず原始プログラムの不良部をどれだけよく指摘してくれるかというエラーチェックないしはメッセージの親切さも評価の対象となる．言語の機能面についていうと，多くのユーザは FORTRAN で機能の多いことを喜ぶ傾向がある．ミニコンの FORTRAN でも JIS 水準 7000 でないと欠点のように

言うし，これに迎合してハードウェア不相応な FORTRAN を売り物にするメーカもあるようだ．ハードウェアとのバランスの良さも評価のうちに加えたい．

ソフトウェアの硬化が叫ばれているが，ソフトネスを二通りの意味で何らかの尺度で測れるとよい．

一つは柔軟な可変性で，追加機能を加えるとか，目的によって手直しの行い易さと，その際に虫を生み付けないような配慮で，近頃はソフトウェアはハードウェアよりハードだ，などとよく言われるのは低い評価ということであろう．

もう一つは，使用者の不注意による正しくない用法に対しての防護という面でシステムの暴走破壊も困るが，勝手に動作して使用者には知らせないで一見もっともらしく動くのも困る．妙に義理堅いのはソフトネスの欠除であろう．

ソフトウェアは人間がコンピュータを使う場合の通路として，人間とハードウェアの間の取り持ちという役目がうまく果せないと困る．特技的に使えば効果が出るのでは駄目で，人間的に使い易いということも大きいポイントであろう．このためにはハードウェアをよく知らない者に不自然に見える制限事項は望ましくない．少くとも外見は人間向きになるようにするのもソフトウェアの設計目標であろう．かつてFORTRANでは数式が使えるとか，COBOL では英語が使えるなどと安易に評価するのを聞いたが，これはどうも買い被りで，数式として正しいもの，英語であれば何でも，という訳ではないので，却って非人間的になっているという批評もあろう．評価方針や手法，道具の確立が望まれる．

改良の余地が残されていることが欠点かどうか？という点については，先程の第1のソフトネスと共に改善指針の得易い配慮はこの一見欠点である余地の裏打ちとなろう．

言語やアプリケーションと共にハードウェアもシステムとしたセットとしてのソフトウェアの評価がより現実的であろう．言語機能という点で東北大学・大型計算機センタでの例として ALGÖL について付言する．

相当高水準（JIS 水準 7060+α−β）の ALGÖL があるが，使用者は数パーセント以下で誰かが雑草と評した FORTRAN の使用者が殆どである．これはこのセンタについていう限り FORTRAN の評価が ALGOL を上回っているということではなかろうか．

大学計算センタにおけるソフトウェア評価

石田晴久

大学の計算機センタのひとつである東大大型計算機センタでは，ソフトウェアの評価は，現用のシステム (HITAC 8800/8700) の改良と将来のシステムの設計とのための資料をうる目的で行っている．その際センタには 2,000 人以上の研究者ユーザがいるところから，評価の重点は

各種ソフトウェアの利用頻度
実行速度（長時間ジョブが多いため．コンパイルは多少遅くてもよい）
エラー・チェック機能(徹底しているのがよい)
使い勝手のよさ（とくに TSS で問題となるが，定量的な方法はない）
信頼性（虫のないこと）

これらの項目を評価する手段としては次のような方法をとっている．

- アカウント・データの詳細な分析
- ソフトウェアの使用頻度カウンタの埋込み
- ベンチ・マーク・テスト
- グラフィック・モニタリング

まずアカウント・データとしては，東大システムの OS (OS 7 と呼ぶ) では，各ジョブごとに 448 バイトもの大量な情報が出るので，データは豊富にあり，いろいろな分析ができる．**表-1** は 1974 年 10 月の全ジョブの中で各言語プロセッサおよびユーティリティを使うジョブの割合である．これでみると，いろいろ非難はあってもわれわれのセンタでは FORTRAN が最も重要な言語（実用的な意味で最良の言語）であり，そのコンパイラをよくすることが大切なことが分る．なお最適化については OPT=0（とくにしない）が 41%，OPT=1（中位），OPT=2（徹底してやる）が 58% だから，OPT=1 のレベルは必要ないといって

表-1　言語プロセッサとユーテリティの使われ方

		（カッコ内は比重）
FORTRAN	76%	(96%)
COBOL	1.4%	(2%)
PL/I	1.3%	(1.6%)
BASIC (TSS 用)	0.1%	(0.1%)
アセンブラ	0.5%	(0.6%)
リンケージ・エディタ	62%	(74%)
エディタ	9%	(11%)
データ・エディタ	6%	(7%)
デバッグ支援	5%	(6%)
オブジェクト・エディタ	1%	(1%)

よい．一方 TSS ジョブ（全体の 9%）のみをとれば，利用頻度は，FORTRAN コンパイラ 24% に対し，エディタ（ソース・プログラムの）は 48%，データ・エディタは 21% になっており，当然ながらエディタが非常に重要なことが裏書きされる．

次に使用頻度カウンタの方は，ライブラリ・プログラムを使用するのに用いるマクロコマンドに埋込んであり，ライブラリ・プログラムの使用頻度がいろんな使い方（コンパイル，実行，コピー，印刷）ごとにとれるようになっている．使用頻度の多いのは，XY プロッタ，連立1次方程式，固有値，逆行列，代数方程式，数値積分，ベッセル関数，乱数などであり，改良や保守はこの辺を重点的にやればよいことが分る．

一方ソフトウェアのスピード（やシステムのスループット）を測るためには，ベンチマーク・テストが欠かせないが，HITAC 8800/8700 のようにバッファ・メモリ (cache) や仮想記憶をもつシステムでは，スピードはプログラムの局所性の高低にかなり左右されるので，スピードの評価は非常に難しい．たとえば8700と 8800 のスピードの比は普通の FORTRAN ジョブの実行で 1：3 前後であるが，いろいろなプログラムについて調べてみると，1：2 から 1：12（バッファ・メモリの容量差がきくとき）にも及ぶ．今後のハードウェアで演算速度の大幅な向上が望めない現在，言語プロセッサではオブジェクトの実行速度を上げることが重要だから，実行速度の評価はやはり重点項目である．

手段の最後にあげたグラフィック・モニタリングは目下開発中のソフトウェア・モニタによるもので，それで収集した情報をグラフィック・ディスプレイでみようとするものである．これは，多重プロセッシングを行う4台の CPU のアイドル時間やメモリ使用状況などを表示するほか，OS 7 の内部のタスクの動きや，リエントラント・プログラムが本当に共用される割合などを調べることを目的としている．後者のリエントラント・プログラムの共用される確率などはこれまで明らかにされていないので，明らかになれば，プログラムをリエントラントにすることの意義が明確に論議できるようになると期待される．

ユーザの立場からみたソフトウェアの評価

石 崎 純 夫

1. アプリケーションによって同じソフトウェアでも評価は異なる

ソフトウェアの評価は，アプリケーションによって，同じものでも評価が異なってくることに注意しなければならない．

〈例1〉 われわれの方では，オンライン・リアルタイム・システムのプログラムはすべてアセンブラで，バッチ・システムのプログラムはすべてコンパイラで組んでいる．何故ならば，9百万口座のファイル・メンテナンス，1日2百万件ものトランザクション処理にあたっては，コンパイラのもつ数々の長所を犠牲にしてでも，きめ細かいプログラムを組むことによるパーフォマンスの向上を採らざるをえないからである．

〈例2〉 9百万口座のファイル・毎日のメンテナンスには，いわゆる汎用のデータ・ベース管理言語はスループットの点で問題があり使えない．しかしながら GIS のような汎用 DBMS は，19,000 人の人事情報システムでは頻繁に使われて効果を発揮している．

〈例3〉 TSS は Management Science 部門では，毎日の業務には不可欠の道具となっているが，一般の事務計算では，われわれのところの事務量ではとても経済的に引き合わない．しかしながら，事務量の少ない中小企業では在庫管理などにも使われて十分効果を発揮している．

その意味では，Gibson-mix 法1つ使うにあたってもアプリケーションを十分考慮して利用すべきであるし，アプリケーションいかんでは，文字通り「猫に小判」ということもありうるのである．

2. 汎用性の限界

プログラムが汎用性をねらえばねらう程オーバヘッドが増し，スループットを減少させ，場合によってはコア常駐を増加させてマルチプログラミングの効率を減少させる場合もあることに注意しなければならない．

〈例1〉 某社の JCL は，あまりにも汎用性をねらいすぎて，使いづらいことには定評がある．

〈例2〉 汎用のデータ・ベース管理システムにおい

て，ネットワーク構造などの扱えるのはよいが，簡単なトリー構造などの処理効率がきわめて悪いものが多い．今むしろ大多数のユーザが望んでいるものは，ネットワークのような複雑な構造を扱えなくても，簡単な構造の大量データを効率よく扱える汎用のデータ・ベース管理システムではあるまいか．

〈例3〉 その意味でも closed 型と open 型の DBMS を一本化しようという試みは，パーフォマンスの点でいかがなものであろうか．Carnegie Mellon 大学の C. Kriebel 教授の「トラック・オートバイ・レーシングカーは，それぞれが高い性能で個別の目的に見合うよう設計された独自のエンジンを有している．ひとつのパッケージであらゆる人々にあらゆることを提供しようというコンピュータシステムや MIS は，出発点

```
RUN ***STATPACK
STATPACK    11:46   10/14/74  MONDAY     IBM

ARE YOU A STATPACK EXPERT
?* NO

SPECIFY THE NAMES OF THE INPUT AND OUTPUT FILES(FORM:IN,OUT)
?* EPAOUT,*

WHAT ANALYSIS DO YOU WISH TO PERFORM
?* SOS

THE FOLLOWING ANALYSES ARE AVAILABLE ...

  EDIT & MISC. OPERATIONS.. EDIT, READ, SAVE, PRINT DATA
  TRANSFORMATION.......... LOG, SQUARE ROOT, ETC.
  ELEMENTARY STATISTICS.... MEAN, STANDARD DEVIATION, ETC.
  CORRELATION
  CROSS TABULATION......... FREQUENCY TABLE FOR 2 CROSSED VARIABLES
  SCATTER DIAGRAM
  HISTOGRAM
  LINE PLOT................ VERTICAL LINE PLOT
  GRAPH.................... PREPARES A FILE FOR **PLOTTER
  RANK CORRELATION
  CHI-SQUARE............... CONTINGENCY TABLE
  T TEST................... TWO SAMPLE MEANS
  REGRESSION............... ONE INDEPENDENT VARIABLE
  STEP-WISE REGRESSION
  MULTIPLE REGRESSION...... WITHOUT STEP-WISE FEATURE
  POLYNOMIAL REGRESSION
  ANALYSIS OF VARIANCE
  CANONICAL CORRELATION
  FACTOR ANALYSIS.......... PRINCIPAL COMPONENT AND VARIMAX ROTATION
  DISCRIMINANT ANALYSIS.... MULTIPLE GROUPS
  EXPONENTIAL SMOOTHING.... TRIPLE EXPONENTIAL
  PROBIT ANALYSIS
  SPECTRAL ANALYSIS
  F TEST................... TWO SAMPLE VARIANCES
  VARIABLE DIMENSIONING.... UP TO 30 COLUMNS MAXIMUM
  SAVE DATA................ ON AN OUTPUT FILE
     (TYPE FINISH TO END THE PROGRAM)
?* TRANSFORMATION

DO YOU WISH TO ENTER DATA FROM YOUR INPUT FILE NOW
?* YES

155 ROWS,  5 COLUMNS (FILE:EPAOUT     )

LAST DATA ITEM READ FROM INPUT FILE EPAOUT

PRINT (REPLY SOS,YES,NO OR ALL)
?* NO
```

① TSS 利用者が非専門家の場合

```
RUN ***STATPACK

STATPACK    13:22   10/14/74  MONDAY     IBM

ARE YOU A STATPACK EXPERT
?* YES

FILE NAMES
?* EPAOUT,*

ANALYSIS
?* TR

READ FILE
?* YES

155 ROWS,  5 COLUMNS (FILE:EPAOUT     )

LAST DATA ITEM READ FROM INPUT FILE EPAOUT
```

② TSS 利用者が専門家の場合

図-1

においてすでに失敗を運命づけられている」という言葉がここにもあてはまらないことを切望するものである．

3. プログラムのモジュール化

イタリヤのあるユーザでは，128kB の GIS のうち IR 部分だけを 20kB に圧縮して取り出し，日中人事部員が何時でも EDP 部門に気兼ねなく CRT に映し出せるようにしていた．OS についても，できるだけモシュール化を進め，ユーザが必要部分のみを組合わせ使えたらどんなに有難いかと思う．

4. Non-EDP 向けのソフトウェア

私は，TSSの最大の利点は，コンピュータのバックグラウンドが全くない人でもコンピュータを使えるところにあると考えている．

〈例1〉 **図-1** は，専門家と素人とで，いかに同じプログラムがブランチしていくかの一例であり，こうすることによって使用時間も費用も大幅に軽減が可能である．

〈例2〉 **図-2** は，コンピュータの知識のない経理部員でも翌月の資金尻の予測をし，シミュレーションができるようにと，われわれのところで開発し，使われているシステムの一例である．

〈例3〉 FORTRAN で 1,000 ステップかかったプログラムを，汎用の TSS アプリケーション・パッケージで組んでみたら 300 ステップででき上がった．

要するに，これからは従来のような一握りのテクノクラートの独占物からコンピュータを解放し，広く

```
RUN ***

/**          14:42JST    01/06/75

1-ｼﾞｺﾖｿｸ  2-ﾀﾞｯｼﾞ ﾐｺﾐ ﾖｿｸ  3-4ﾊﾝﾁ ﾐｺﾐ ﾖｿｸ  4-ﾊﾝﾁ ﾐｺﾐ ﾖｿｸ ---?1
1-1ｯﾁﾖｿｸ  2-ｵｵﾍﾞｯﾖｿｸ  3-ｼﾞｸｲﾚﾗﾖｿｸ---?2
1-ﾀﾞｯｼﾞ  2-4ﾊﾝﾁ  3-ﾊﾝﾁｰ---?1
ﾖｿｸ ｷﾝｺｵ  1-ﾌｼﾞ,2-ｽﾐﾄﾓ,3-ﾜﾝﾜ,4-ﾐｯﾋﾞｼ,5-D.K.B.,6-ﾄｷﾜ---?1
ﾖｿｸ ﾂｷ (OR ｷﾝ)---?9
ﾖｿｸ ﾀｲｼｭｳ ﾈﾝ---?49
ﾊﾟﾗﾒｰﾀｰ ﾍﾝｺｳ ﾊ (1-YES,2-NO)---?2
 R,S  ﾉﾐ ｴﾗﾏｽｶ (1-YES,2-NO)---?1


          R= 0.947                          R= 0.818
          S=  106.                          S=  158.
ｸﾞﾗﾌ ｵ ｴﾗﾏｽｶ (1-YES,2-NO)---?1

1-ﾍｲﾜﾝ,2-ﾏｯﾀﾝ,3-BOTH---?3
SET
```

図-2 富士銀行資金尻予測モデル

EDP のバックグラウンドのない一般社内ユーザにもコンピュータを気楽に使えるようにすることが重要である．TSSの会話用言語は，そのためにも大きく役立っているのである．

報告をおわるにあたって

パネル討論の内容は文章にすると長いものになるので，以上のような形式で圧縮し編集したが，当然のことながら，当日の討論の内容そのものでなく，省かれたり付加されたこともかなりあり，発言の順序も本報告では一部入れかえてある．また，当日の討論が行われなかったことの中に，重要な事項が多くある．それについては，計画や司会の責に帰せられるところが多いと考えている．

なお，本文の編集には，東大大型計算機センタの石田晴久助教授の御助力をいただいた．厚く謝意を表する次第である．（大野）

（昭和50年3月6日受付）